Optimal Scaling of A Gradient Method for Distributed Resource Allocation [*]

Lin Xiao[†] Stephen Boyd[‡]

Revised February 15, 2005

Abstract

We consider a class of weighted gradient methods for distributed resource allocation over a network. Each node of the network is associated with a local variable and a convex cost function; the sum of the variables (resources) across the network is fixed. Starting with a feasible allocation, each node updates its local variable in proportion to the differences between the marginal costs of itself and its neighbors. We focus on how to choose the proportional weights on the edges (scaling factors for the gradient method) to make this distributed algorithm converge, and how to make the convergence as fast as possible.

We give sufficient conditions on the edge weights for the algorithm to converge monotonically to the optimal solution; these conditions have the form of a linear matrix inequality. We give some simple, explicit methods to choose the weights that satisfy these sufficient conditions. We also derive a guaranteed convergence rate for the algorithm, and find the weights that minimize this rate by solving a semidefinite program. Finally, we extend the main results to problems with general equality constraints, and problems with block separable objective function.

Key words: distributed optimization, resource allocation, weighted gradient method, convergence rate, semidefinite programming.

[*]To appear in *Journal of Optimization Theory and Applications*, vol. 129, no. 3, 2006.

[†]Center for the Mathematics of Information, Mail Code 136-93, California Institute of Technology, Pasadena, CA 91125-9300. Email: `lxiao@caltech.edu`.

[‡]Department of Electrical Engineering, Stanford University, Stanford, CA 94305-9510. Email: `boyd@stanford.edu`.

1 Introduction

We consider an optimal resource allocation problem over a network of autonomous agents. The network is modeled as a directed graph $(\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V} = \{1, \dots, n\}$ and edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Each edge (i, j) is an ordered pair of distinct nodes. We define $\mathcal{N}_i$, the set of (oriented) neighbors of node i, as $\mathcal{N}_i = \{j \mid (i, j) \in \mathcal{E}\}$ (in other words, $j \in \mathcal{N}_i$ if there is an edge from node i to node j).

With node i we associate a variable $x_i \in \mathbf{R}$ and a corresponding convex cost function $f_i : \mathbf{R} \to \mathbf{R}$. We consider the following optimization problem:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n f_i(x_i) \\ \text{subject to} & \sum_{i=1}^n x_i = c, \end{array} \tag{1}$$

where $c \in \mathbf{R}$ is a given constant. We can think of x_i as the amount of some resource located at node i, and interpret $-f_i$ as the local (concave) utility function. The problem (1) is to find an allocation of the resource that maximizes the total utility $\sum_{i=1}^n -f_i(x_i)$. In this paper, we are interested in *distributed* algorithms for solving this problem, where each node is only allowed to communicate with its neighbors and conduct local computation. Thus the *local information* structure imposed by the graph should be considered as part of the problem formulation. This simple model for distributed resource allocation and its variations have many applications in economic systems, *e.g.*, [AH60, Hea69], and distributed computer systems [KS89].

We assume that the functions f_i are convex and twice continuously differentiable with second derivatives that are bounded below and above:

$$l_i \leq f_i''(x_i) \leq u_i, \quad x_i \in \mathbf{R}, \quad i = 1, \dots, n, \tag{2}$$

where $l_i > 0$ and u_i are known (the functions are strictly convex). Let $x = (x_1, \dots, x_n) \in \mathbf{R}^n$ denote the vector of the variables and $f(x) = \sum_{i=1}^n f_i(x_i)$ denote the objective function. We use $f^\star$ to denote the optimal value of this problem; *i.e.*, $f^\star = \inf\{f(x) \mid \mathbf{1}^T x = c\}$, where $\mathbf{1}$ denotes the vector with all components one. Under the above assumption, the convex optimization problem (1) has a unique optimal solution $x^\star$. Let $\nabla f(x) = (f_1'(x_1), \dots, f_n'(x_n))$ denote the gradient of f at x. The optimality conditions for this problem are

$$\mathbf{1}^T x^\star = c, \qquad \nabla f(x^\star) = p^\star \mathbf{1}, \tag{3}$$

where $p^\star$ is the (unique) optimal Lagrange multiplier.

In a centralized setup (*i.e.*, all functions and their derivatives can be evaluated at a central agent), many methods can be used to solve the problem (1), or equivalently, the optimality conditions (3). If the functions f_i are all quadratic, the optimality conditions (3) are a set of linear equations in $x^\star$ and $p^\star$, and can be solved directly. In the more general case, the problem can be solved by iterative methods, *e.g.*, the projected gradient method, Newton's method, quasi-Newton methods (*e.g.*, BFGS method), and many others. Detailed accounts of these algorithms (and others) can be found in, *e.g.*, [Ber99] and [BV03].

The design of decentralized mechanisms for resource allocation has a long history in economics [Hur73], and there are two main classes of mechanisms: price-directed [AH60] and resource-directed [Hea69]. However, most of the methods are not *fully* distributed because they either need a central price coordinator or need a central resource dispatcher. So they cannot be applied to the problem we consider. In this paper we will focus on a class of *center-free* algorithms first proposed in [HSS80].

1.1 The center-free algorithm for resource allocation

Assume that we have an initial allocation of the resource $x(0)$ that satisfies $\mathbf{1}^T x(0) = c$. The center-free algorithm for solving problem (1) has the following iterative form:

$$x_i(t+1) = x_i(t) - W_{ii} f_i'(x_i(t)) - \sum_{j \in \mathcal{N}_i} W_{ij} f_j'(x_j(t)), \quad i = 1, \ldots, n, \tag{4}$$

for $t = 0, 1, \ldots$. In other words, at each iteration, each node computes the derivative of its local function, queries the derivative values from its neighbors, and then updates its local variable by a weighted sum of the values of derivatives. Here W_{ii} is the self-weight at node i, and W_{ij} ($j \in \mathcal{N}_i$) is the weight associated with the edge $(i, j) \in \mathcal{E}$. Setting $W_{ij} = 0$ for $j \notin \mathcal{N}_i$, this algorithm can be written in vector form as

$$x(t+1) = x(t) - W \nabla f(x(t)), \tag{5}$$

where $W \in \mathbf{R}^{n \times n}$ is the weight matrix. Thus the center-free algorithm can be thought of as a weighted gradient descent method, in which the weight matrix W has a sparsity constraint given by the graph:

$$W \in \mathcal{S} = \{Z \in \mathbf{R}^{n \times n} \mid Z_{ij} = 0 \text{ if } i \neq j \text{ and } (i, j) \notin \mathcal{E}\}. \tag{6}$$

Throughout this paper we focus on the following question: *How should we choose the weight matrix W?*

We first consider two basic requirements on W. First, we require that all iterates $x(t)$ of the algorithm are feasible, *i.e.*, satisfy $\mathbf{1}^T x(t) = c$ for all t. With the assumption that $x(0)$ is feasible, this requirement will be met provided the weight matrix satisfies

$$\mathbf{1}^T W = 0, \tag{7}$$

since we then have

$$\mathbf{1}^T x(t+1) = \mathbf{1}^T x(t) - \mathbf{1}^T W \nabla f(x(t)) = \mathbf{1}^T x(t).$$

We will also require, naturally, that the optimal point $x^\star$ is a fixed point of the algorithm (5), *i.e.*,

$$x^\star = x^\star - W \nabla f(x^\star) = x^\star - p^\star W \mathbf{1}.$$

This will hold in the general case (with $p^\star \neq 0$) provided

$$W\mathbf{1} = 0. \tag{8}$$

The requirements (7) and (8) show that the vector $\mathbf{1}$ must be both a left and right eigenvector of W, associated with the eigenvalue zero. One special case of interest is when the weight matrix W is symmetric. In this case, of course, the requirements (7) and (8) are the same, and simply state that $\mathbf{1}$ is in the nullspace of W.

Assuming the weights satisfy (8), we have $W_{ii} = -\sum_{j\in\mathcal{N}_i} W_{ij}$, which can be substituted into equation (4) to get

$$x_i(t+1) = x_i(t) - \sum_{j\in\mathcal{N}_i} W_{ij}\left(f_j'(x_j(t)) - f_i'(x_i(t))\right), \quad i = 1,\ldots,n. \tag{9}$$

Thus, the change in the local variable at each step is given by a weighted sum of the differences between its own derivative value and those of its neighbors. The equation (9) has a simple interpretation: at each iteration, each connected pair of nodes shifts resources from the node with higher marginal cost to the one with lower marginal cost, in proportion to the difference in marginal costs. The weight $-W_{ij}$ gives the proportionality constant on the edge $(i,j) \in \mathcal{E}$. (This interpretation suggests that the weights on edges should be negative, but we will see examples where a few positive edge weights actually enhance the convergence rate.)

1.2 Previous work

Distributed resource allocation algorithms of the form (9) were first proposed and studied by Ho, Servi and Suri in [HSS80]. They considered an undirected graph with symmetric weights on the edges, and called algorithms of this form *center-free* algorithms. ('Center-free' refers to the absence of a central coordinating entity.) In the notation of this paper, they assumed $W = W^T$ and $W\mathbf{1} = 0$, and derived the following additional conditions on W that are sufficient for the algorithm (9) to converge to the optimal solution $x^\star$:

$$\begin{array}{ll} (a) & W \text{ is irreducible} \\ (b) & W_{ij} \leq 0, \quad (i,j) \in \mathcal{E} \\ (c) & \sum_{j\in\mathcal{N}_i} |W_{ij}| < 1/u_{\max}, \quad i = 1,\ldots,n \end{array} \tag{10}$$

where $u_{\max}$ is an upper bound on the second derivatives of the functions f_i, *i.e.*, $u_{\max} \geq \max_i u_i$. The first condition, that W is irreducible, is equivalent to the statement that the subgraph consisting of all the nodes and edges with nonzero weights is connected. We will show that these conditions are implied by those established in this paper.

It should be noted that the problem considered in [HSS80] has nonnegativity constraints on the variables: $x_i \geq 0$, $i = 1,\ldots,n$ (with $c > 0$). They gave a separate initialization procedure, which identifies and eliminates some nodes that will have zero value at optimality (not necessarily all such nodes). As a result of this initialization procedure and some additional conditions on the initial point $x(0)$, all following iterates of the center-free algorithm (9) automatically satisfy the nonnegativity constraints. In [KS89], second derivatives of the functions f_i are used to modify the algorithm (9) (with a constant weight on all edges) to obtain faster convergence. An interesting analogy between various iterative algorithms for solving problem (1) and the dynamics of several electrical networks can be found in [Ser80].

Many interesting similarities exist between the resource allocation problem and network flow problems with convex separable cost (see, *e.g.*, [Roc84, BT89, Ber98] and references therein). In particular, by ignoring the local information structure, problem (1) can be formulated as a simple network flow problem with two nodes and n links connecting them. Thus many distributed algorithms for network flow problems such as those in [TB86, Baz96] can be used; see also [LT94] for a convergence rate analysis of such an algorithm. However, with the imposed local information structure on a graph, the resource allocation problem has a distinct nature, and the above mentioned algorithms cannot be applied directly. The center-free algorithm considered in this paper belongs to a more general class of gradient-like algorithms studied in [TBA86].

In this paper, we give weaker sufficient conditions than (10) for the center-free algorithm to convergence, and optimize the edge weights to get fast convergence. Our method is closely related to the approach in [BDX03], where the problem of finding the fastest mixing Markov chain on a graph is considered. In [XB03], the same approach was used to find fast linear iterations for a distributed average consensus problem.

1.3 Outline

In §2, we give sufficient conditions on the weight matrix W under which the algorithm (5) converges to the optimal solution monotonically. These conditions involve a linear matrix inequality (LMI) in the weight matrix. Moreover, we quantify the convergence by deriving a guaranteed convergence rate for the algorithm. In §3, we give some simple, explicit choices for the weight matrix W that satisfy the convergence conditions. In §4, we propose to minimize the guaranteed convergence rate obtained in §2 in order to get fast convergence of the algorithm (5). We observe that the optimal weights (in the sense of minimizing the guaranteed convergence rate) can be found by solving a semidefinite program (SDP). In §5, we show some numerical examples that demonstrate the effectiveness of the proposed weight selection methods. Finally, in §6, we extend the main results to problems with general equality constraints, and problems with block separable objective functions. We give our conclusions and some final remarks in §7.

2 Convergence conditions

In this section, we state and prove the main theorem. We use the following notation: L and U denote diagonal matrices in $\mathbf{R}^{n\times n}$ whose diagonal entries are the lower bounds l_i and upper bounds u_i given in (2). Note that L and U are positive definite. For a symmetric matrix Z, we list its eigenvalues (all real) in nonincreasing order, as $\lambda_1(Z) \geq \lambda_2(Z) \geq \cdots \geq \lambda_n(Z)$, where $\lambda_i(Z)$ denotes the ith largest eigenvalue of Z.

Theorem 1 *If the weight matrix W satisfies $\mathbf{1}^T W = 0$, $W\mathbf{1} = 0$, and*

$$\lambda_{n-1}\left(L^{1/2}(W + W^T - W^T U W)L^{1/2}\right) > 0, \tag{11}$$

then the algorithm (5) converges to the optimal solution $x^\star$ of problem (1), and the objective function decreases monotonically. In fact, we have

$$f(x(t)) - f^\star \le \eta(W)^t (f(x(0)) - f^\star) \tag{12}$$

with a guaranteed convergence rate

$$\eta(W) = 1 - \lambda_{n-1}\left(L^{1/2}(W + W^T - W^T U W)L^{1/2}\right). \tag{13}$$

Moreover, the condition (11) is equivalent to the strict linear matrix inequality (LMI)

$$\begin{bmatrix} W + W^T + (1/n)\mathbf{1}\mathbf{1}^T & W \\ W^T & U^{-1} \end{bmatrix} \succ 0. \tag{14}$$

(Here a matrix $A \succ 0$ means that A is positive definite.)

Proof.

Let $\Delta x(t)$ denote $x(t+1) - x(t)$. Using the algorithm (5), we have

$$\Delta x(t) = -W \nabla f(x(t)).$$

The Taylor expansion of f_i at $x_i(t)$ yields

$$f_i(x_i(t+1)) = f_i(x_i(t)) + f_i'(x_i(t))\Delta x_i(t) + \frac{1}{2} f_i''(z_i(t))\Delta x_i(t)^2$$

where $z_i(t)$ is between $x_i(t)$ and $x_i(t+1)$. Let $\nabla^2 f(x) = \mathbf{diag}(f_1''(x_1), \ldots, f_n''(x_n))$ be the Hessian matrix of f at x. The objective function at iterate $t+1$ can be written as

$$\begin{aligned} f(x(t+1)) &= f(x(t)) + \nabla f(x(t))^T \Delta x(t) + \frac{1}{2}\Delta x(t)^T \nabla^2 f(z(t)) \Delta x(t) \\ &= f(x(t)) - \nabla f(x(t))^T W \nabla f(x(t)) + \frac{1}{2}\nabla f(x(t))^T W^T \nabla^2 f(z(t)) W \nabla f(x(t)) \\ &= f(x(t)) - \frac{1}{2}\nabla f(x(t))^T \left(W + W^T - W^T \nabla^2 f(z(t)) W\right) \nabla f(x(t)). \end{aligned}$$

Using the assumption (2), we have $\nabla^2 f(z(t)) \preceq U$, so

$$\begin{aligned} f(x(t+1)) &\le f(x(t)) - \frac{1}{2}\nabla f(x(t))^T \left(W + W^T - W^T U W\right) \nabla f(x(t)) \\ &= f(x(t)) - \frac{1}{2}\nabla f(x(t))^T L^{-1/2} V L^{-1/2} \nabla f(x(t)), \end{aligned} \tag{15}$$

where

$$V = L^{1/2}\left(W + W^T - W^TUW\right)L^{1/2}.$$

From conditions (7) and (8), *i.e.*, $\mathbf{1}^TW = 0$ and $W\mathbf{1} = 0$, we conclude that $L^{-1/2}\mathbf{1}$ is an eigenvector of the symmetric matrix V associated with the eigenvalue zero. Let $e(t)$ be the projection of $L^{-1/2}\nabla f(x(t))$ onto the subspace that is orthogonal to $L^{-1/2}\mathbf{1}$:

$$\begin{aligned} e(t) &= \left(I - \frac{1}{\mathbf{1}^TL^{-1}\mathbf{1}}L^{-1/2}\mathbf{1}\mathbf{1}^TL^{-1/2}\right)L^{-1/2}\nabla f(x(t)) \\ &= L^{-1/2}\left(\nabla f(x(t)) - \frac{\mathbf{1}^TL^{-1}\nabla f(x(t))}{\mathbf{1}^TL^{-1}\mathbf{1}}\mathbf{1}\right). \end{aligned}$$

Replacing $L^{-1/2}\nabla f(x(t))$ by $e(t)$ does not change the equation (15), so we have

$$f(x(t+1)) \le f(x(t)) - \frac{1}{2}e(t)^TVe(t). \tag{16}$$

Now the condition (11), *i.e.*, $\lambda_{n-1}(V) > 0$ means that the matrix V is positive semidefinite and $\lambda_n(V) = 0$ is a simple eigenvalue (with associated eigenvector $L^{-1/2}\mathbf{1}$). This implies that $f(x(t+1)) < f(x(t))$, *i.e.*, that the objective function decreases provided $e(t) \neq 0$ When $e(t) = 0$, the gradient $\nabla f(x(t))$ must be a multiple of $\mathbf{1}$, *i.e.*, the optimality conditions (3) are satisfied; in this case, we conclude that $x(t)$ is optimal.

Next we derive the guaranteed convergence rate (13). Note that with the condition (11) and the inequality (16), we have

$$f(x(t+1)) - f(x(t)) \le -\frac{1}{2}\lambda_{n-1}(V)\|e(t)\|^2. \tag{17}$$

We shall derive another inequality relating $f(x(t))$, $f^\star$ and $\|e(t)^2\|$. To do so, we again use the Taylor expansion of f at $x(t)$. Using the assumption (2), we have

$$f(y) \ge f(x(t)) + \nabla f(x(t))^T(y - x(t)) + \frac{1}{2}(y - x(t))^TL(y - x(t))$$

for any $y \in \mathbf{R}^n$. If y is feasible (true for any iterate of the proposed algorithm), then $\mathbf{1}^T(y - x(t)) = 0$. So we have

$$\begin{aligned} f(y) &\ge f(x(t)) + \left(\nabla f(x(t)) - \frac{\mathbf{1}^TL^{-1}\nabla f(x(t))}{\mathbf{1}^TL^{-1}\mathbf{1}}\mathbf{1}\right)^T(y - x(t)) + \frac{1}{2}(y - x(t))^TL(y - x(t)) \\ &= f(x(t)) + e(t)^TL^{1/2}(y - x(t)) + \frac{1}{2}(y - x(t))^TL(y - x(t)) \\ &= f(x(t)) + e(t)^Tz + \frac{1}{2}z^Tz \end{aligned}$$

where $z = L^{1/2}(y - x(t))$. Minimizing the right-hand side over z yields

$$f(y) \ge f(x(t)) - \frac{1}{2}\|e(t)\|^2.$$

Since this is true for all feasible y, it is certainly true for $x^\star$. In other words, we have

$$f^\star \geq f(x(t)) - \frac{1}{2}\|e(t)\|^2. \tag{18}$$

Now combining the inequalities (17) and (18) yields

$$f(x(t+1)) - f^\star \leq (1 - \lambda_{n-1}(V))\,(f(x(t)) - f^\star) \tag{19}$$

which gives the desired result (12) and (13). Since all the iterates $x(t)$ are feasible, we always have

$$f(x(t)) - f(x(t+1)) \leq f(x(t)) - f^\star.$$

Applying the two inequalities (17) and (18) on two sides of the above inequality respectively yields $\lambda_{n-1}(V) \leq 1$, hence $0 < \eta(W) < 1$. It follows that

$$\lim_{t\to\infty} f(x(t)) = f^\star,$$

i.e., the algorithm converges to the optimal solution under condition (11).

It remains to show that the eigenvalue condition (11) is equivalent to the strict LMI (14). Since L is diagonal and positive definite, $\lambda_{n-1}(V) > 0$ if and only if

$$\lambda_{n-1}(W + W^T - W^T U W) > 0,$$

which can be expressed as the quadratic matrix inequality

$$W + W^T - W^T U W + (1/n)\mathbf{1}\mathbf{1}^T \succ 0.$$

Here the rank-one matrix $(1/n)\mathbf{1}\mathbf{1}^T$ has its only nonzero eigenvalue one associated with the eigenvector $\mathbf{1}$. Finally, using Schur complements, the above quadratic matrix inequality is equivalent to the LMI (14). Note that we do not need to know L to test the convergence condition (11), although it is needed in calculating the guaranteed convergence rate. $\square$

Remark The derivation of the equation (19) holds without assuming the convergence condition (11), so long as we interpret $\lambda_{n-1}(V)$ as the smallest eigenvalue of V excluding the zero eigenvalue associated with the eigenvector $\mathbf{1}$. It is evident from (19) that $\lambda_{n-1}(V) > 0$ is necessary for $\eta(W) < 1$, and sufficient for the algorithm to converge to the optimal solution.

2.1 Conditions for symmetric weights

When the weight matrix W is symmetric, the convergence conditions reduce to

$$W = W^T, \quad W\mathbf{1} = 0 \tag{20}$$
$$2W + (1/n)\mathbf{1}\mathbf{1}^T \succ 0 \tag{21}$$
$$2U^{-1} - W \succ 0. \tag{22}$$

To see this, we first rewrite the LMI (14) for symmetric W:

$$\begin{bmatrix} 2W + (1/n)\mathbf{1}\mathbf{1}^T & W \\ W & U^{-1} \end{bmatrix} \succ 0.$$

Applying Schur complements, this LMI is equivalent to (21) and

$$U^{-1} - W\left(2W + (1/n)\mathbf{1}\mathbf{1}^T\right)^{-1} W \succ 0.$$

Under the conditions (20) and (21), we have

$$W\left(2W + (1/n)\mathbf{1}\mathbf{1}^T\right)^{-1} W = (1/2)W.$$

So the above inequality is precisely (22).

The conditions (20)-(22) are generalizations of (10). In particular, we consider the conditions that the matrix $2U^{-1} - W$ is strictly diagonally dominant, which is sufficient for (22) to hold. The strictly diagonal dominance property can be expressed as

$$\left|\frac{2}{u_i} - W_{ii}\right| > \sum_{j\in\mathcal{N}_i} |W_{ij}|, \quad i = 1, \ldots, n.$$

If all the off-diagonal elements of W are nonpositive and $W_{ii} = -\sum_{j\in\mathcal{N}_i} W_{ij}$, the above condition becomes

$$\sum_{j\in\mathcal{N}_i} |W_{ij}| < \frac{1}{u_i}, \quad i = 1, \ldots, n. \tag{23}$$

When all the numbers u_i are equal (or simply take their maximum), as assumed in [HSS80], the inequality (23) is exactly the third condition in (10).

3 Simple weight selections

In this section, we give two simple methods to select the weight matrix so that it satisfies the convergence conditions established in the previous section. Here we only consider symmetric weight matrices, and the methods are based on the sufficient condition (23). For symmetric weight matrices, each edge of the graph is bidirectional and has the same weight in both directions, so each can be considered as an undirected edge with a single weight.

3.1 Constant weight on the edges

The simplest and most commonly used method is to have constant weight on all the edges of the graph, and obtain the self-weights W_{ii} from the equality constraint $W\mathbf{1} = 0$:

$$W_{ij} = \begin{cases} \alpha & (i,j) \in \mathcal{E} \\ -d_i\alpha & i = j \\ 0 & \text{otherwise,} \end{cases}$$

where $d_i = |\mathcal{N}_i|$ is the degree of node i. From the condition (23), we can deduce a range of α that guarantees convergence of the algorithm:

$$\frac{-1}{\max_{i\in\mathcal{N}} d_i u_i} < \alpha < 0. \tag{24}$$

Actually, it is also safe to set

$$\alpha = \frac{-1}{\max_{i\in\mathcal{N}} d_i u_i} \tag{25}$$

unless all the $d_i u_i$'s are equal. This can be verified by considering the *irreducibly diagonally dominant* property of the matrix $2U^{-1} - W$; see, *e.g.*, [HJ85, §6.2].

We call the constant weight given in (25) the *max-degree* weight. It comes from the fact that when the functions f_i are appropriately scaled such that $u_i = 1$ for all i, then α is determined solely by the maximum degree of the graph. In fact the range (24) is usually very conservative (the constant can be made more negative). In §4.2, we will determine the constant weight that minimizes the guaranteed convergence rate η established in theorem 1. That constant weight is often outside of the range (24).

3.2 Weights determined by local information

A slightly more sophisticated method is to determine a range for the weight on each edge of the graph. From the condition (23), it is straightforward to obtain the following ranges for the edge weights that guarantee the convergence of the algorithm:

$$-\min\left\{\frac{1}{d_i u_i}, \frac{1}{d_j u_j}\right\} < W_{ij} < 0, \quad (i,j)\in\mathcal{E}.$$

As before, after choosing the edge weights in these ranges, we can determine the self-weights using $W_{ii} = -\sum_{j\in\mathcal{N}_i} W_{ij}$. Again, unless all the nodes have the same value of $d_i u_i$, we can always set

$$W_{ij} = -\min\left\{\frac{1}{d_i u_i}, \frac{1}{d_j u_j}\right\}, \quad (i,j)\in\mathcal{E}. \tag{26}$$

We call these weights the *Metropolis* weights, because the main idea of this method relates to the Metropolis algorithms for choosing transition probabilities on a graph to make the associated Markov chain mix rapidly ([MRR$^+$53]; see also, *e.g.*, [DSC98, BDX03]). In §5, we will see that the weights selected by this method often perform better (*i.e.*, make the algorithm (5) converge more rapidly) than the max-degree constant weight on all edges.

Note that this method of choosing edge weights only relies on *local* information in the graph: the degrees and the upper bounds on the second derivatives of the functions at its two incident nodes. This property allows the weights to be selected in a distributed manner, which is ideal for a distributed algorithm running on the graph such as (9).

4 Optimal scaling of the center-free algorithm

In this section we pose the question of how to choose the weight matrix to make the algorithm (5) converge as fast as possible. Clearly the convergence properties depend on the particular objective functions and initial condition. When the lower and upper bounds L and U are the only information available, it is reasonable to choose the weight matrix to minimize the guaranteed convergence rate η established in theorem 1. This is equivalent to maximizing the second smallest eigenvalue of the matrix V, *i.e.*,

$$\begin{array}{ll} \text{maximize} & \lambda_{n-1}\left(L^{1/2}(W+W^T-W^TUW)L^{1/2}\right) \\ \text{subject to} & W \in \mathcal{S}, \quad \mathbf{1}^TW=0, \quad W\mathbf{1}=0, \end{array} \tag{27}$$

where the optimization variable is W. In this section, we show that this problem is convex and can be converted into a semidefinite program (SDP), and so can be efficiently and globally solved using numerical algorithms such as interior-point methods (see, *e.g.*, [VB96]).

In the special case of choosing the constant edge weight to minimize the guaranteed rate, we show that with appropriate scaling of the objective functions, the solution can be directly given in terms of the eigenvalues of the Laplacian matrix of the graph.

4.1 Weight design via SDP

We first show that the eigenvalue optimization problem (27) can be converted to an SDP. Let s be a lower bound on the eigenvalues of $L^{1/2}(W+W^T-W^TUW)L^{1/2}$ on the subspace that is orthogonal to $L^{-1/2}\mathbf{1}$ (which corresponds to the eigenvalue zero). Then we have

$$L^{1/2}(W+W^T-W^TUW)L^{1/2} \succeq s\left(I-\frac{1}{\mathbf{1}^TL^{-1}\mathbf{1}}L^{-1/2}\mathbf{1}\mathbf{1}^TL^{-1/2}\right),$$

which is equivalent to (by multiplying $L^{-1/2}$ on the left and right)

$$W+W^T-W^TUW \succeq s\left(L^{-1}-\frac{1}{\mathbf{1}^TL^{-1}\mathbf{1}}L^{-1}\mathbf{1}\mathbf{1}^TL^{-1}\right). \tag{28}$$

Using Schur complements, the above quadratic matrix inequality is equivalent to the LMI

$$\begin{bmatrix} W+W^T-s\left(L^{-1}-\frac{1}{\mathbf{1}^TL^{-1}\mathbf{1}}L^{-1}\mathbf{1}\mathbf{1}^TL^{-1}\right) & W^T \\ W & U^{-1} \end{bmatrix} \succeq 0. \tag{29}$$

Therefore the eigenvalue optimization problem (27) is equivalent to the SDP

$$\begin{array}{ll} \text{maximize} & s \\ \text{subject to} & W \in \mathcal{S}, \quad \mathbf{1}^TW=0, \quad W\mathbf{1}=0 \\ & \begin{bmatrix} W+W^T-s\left(L^{-1}-\frac{1}{\mathbf{1}^TL^{-1}\mathbf{1}}L^{-1}\mathbf{1}\mathbf{1}^TL^{-1}\right) & W^T \\ W & U^{-1} \end{bmatrix} \succeq 0 \end{array} \tag{30}$$

with optimization variables s and W.

Note that the matrices on both sides of the inequality (28) have the common eigenvalue zero associated with the eigenvector $\mathbf{1}$. As a result, the LMI (29) has empty interior, which can cause trouble for some classes of interior-point methods (see, *e.g.*, [NN94, Ye97, WSV00, BV03]). But this problem is readily avoided: we simply add the rank-one matrix $(1/n)\mathbf{1}\mathbf{1}^T$ to the left hand side of (28) to get the inequality

$$W + W^T - W^T U W + \frac{1}{n}\mathbf{1}\mathbf{1}^T \succeq s\left(L^{-1} - \frac{1}{\mathbf{1}^T L^{-1}\mathbf{1}} L^{-1}\mathbf{1}\mathbf{1}^T L^{-1}\right).$$

This leads to the SDP

$$\begin{array}{ll} \text{maximize} & s \\ \text{subject to} & W \in \mathcal{S}, \quad \mathbf{1}^T W = 0, \quad W\mathbf{1} = 0 \\ & \left[\begin{array}{cc} W + W^T + \frac{1}{n}\mathbf{1}\mathbf{1}^T - s\left(L^{-1} - \frac{1}{\mathbf{1}^T L^{-1}\mathbf{1}} L^{-1}\mathbf{1}\mathbf{1}^T L^{-1}\right) & W^T \\ W & U^{-1} \end{array}\right] \succeq 0. \end{array} \tag{31}$$

This SDP is equivalent to (30), but here the LMI constraint has nonempty interior, so general interior-point methods can be applied to solve this problem.

Since W is often very sparse, exploiting sparsity can further improve the efficiency of numerical algorithms and allow the solution of very large scale problems. We discuss how to exploit sparsity in interior-point methods and a simple subgradient method for solving a similar class of SDPs in [XB03]. The same techniques apply in this case.

4.2 Best constant weight

If the weight matrix is symmetric and all the edge weights are equal to a constant α (as described in §3.1), we can write the weight matrix as

$$W = -\alpha \mathcal{L} \tag{32}$$

where $\mathcal{L}$ is the *Laplacian* matrix of the graph, which is defined as

$$\mathcal{L}_{ij} = \begin{cases} -1 & (i,j) \in \mathcal{E} \\ d_i & i = j \\ 0 & \text{otherwise.} \end{cases}$$

The Laplacian matrix is positive semidefinite. Since the graph is assumed connected, it has a simple eigenvalue $\lambda_n(\mathcal{L}) = 0$ with associated eigenvector $\mathbf{1}$ (see, *e.g.*, [Mer94, GR01]). From the expression (32), we must have $\alpha < 0$ for W to be positive semidefinite (see condition (21)).

We can substitute (32) into the SDP (31) to solve for the best constant $\alpha^\star$. Here we consider a simple case where $\alpha^\star$ can be found analytically. We assume that the functions f_i are appropriately scaled such that the upper bound on the second derivatives is $U = I$ (the identity matrix), and the lower bound is $L = \beta I$ with $0 < \beta < 1$. In this case,

$$V = \beta(W + W^T - W^T U W) = \beta(-2\alpha\mathcal{L} - \alpha^2\mathcal{L}^2) = \beta I - \beta(I + \alpha\mathcal{L})^2.$$

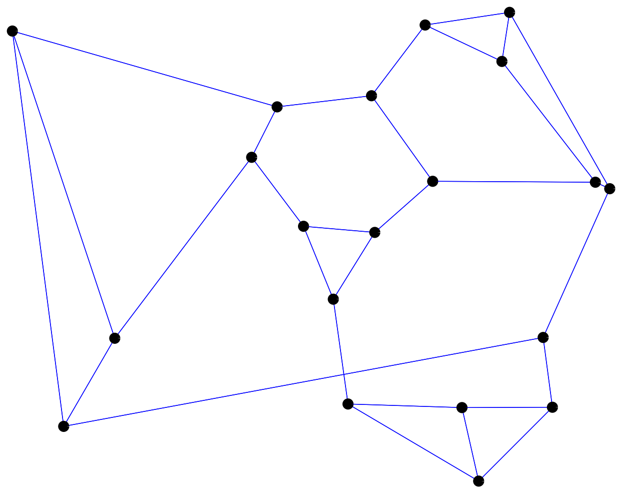

Figure 1: A randomly generated regular graph with 20 nodes and constant degree 3.

Now we can express $\lambda_{n-1}(V)$ in terms of β and two extreme eigenvalues of $\mathcal{L}$:

$$\lambda_{n-1}(V) = \beta - \beta\Big(\max\{1 + \alpha\lambda_{n-1}(\mathcal{L}),\ \ -1 - \alpha\lambda_1(\mathcal{L})\}\Big)^2.$$

For $\lambda_{n-1}(V) > 0$, the weight α must lie in the range

$$\frac{-2}{\lambda_1(\mathcal{L})} < \alpha < 0.$$

The optimal constant weight that maximizes $\lambda_{n-1}(V)$ is

$$\alpha^\star = \frac{-2}{\lambda_1(\mathcal{L}) + \lambda_{n-1}(\mathcal{L})}.$$

It is interesting to note that this result is very similar to the best constant weight found in the fast averaging problem, *cf.* [XB03, §4.1]. The only difference is the sign, which is due to different sign conventions used.

5 A numerical example

In this section, we use simulation to demonstrate the effectiveness of the weight matrices determined by various methods proposed in the previous two sections. We consider the graph shown in Figure 1. This is a randomly generated regular graph with $n = 20$ nodes, where each node has degree three. Each edge is bidirectional. In this example, we will consider both symmetric and nonsymmetric weight matrices.

We use the following family of functions:

$$f_i(x_i) = \frac{1}{2}a_i(x_i - c_i)^2 + \log\left(1 + e^{b_i(x_i - d_i)}\right), \quad i = 1, \ldots, n,$$

method	max-degree	Metropolis	best constant	SDP symm.	SDP nonsymm.
$\eta(W)$	0.9503	0.9237	0.9217	0.8750	0.8729

Table 1: Guaranteed convergence rates of different weight matrices.

with the coefficients a_i, b_i, c_i, d_i generated randomly with uniform distributions on $[0, 2]$, $[-2, 2]$, $[-10, 10]$ and $[-10, 10]$ respectively. The second derivatives of these functions are

$$f_i''(x_i) = a_i + b_i^2 \frac{e^{b_i(x_i-d_i)}}{(1+e^{b_i(x_i-d_i)})^2}, \quad i = 1, \ldots, n,$$

which have the following lower and upper bounds:

$$l_i = a_i, \quad u_i = a_i + \frac{1}{4}b_i^2, \quad i = 1, \ldots, n.$$

We assume that the sum of the variables is fixed to zero, *i.e.*, $\sum_{i=1}^n x_i = 0$.

Using these bounds, we find the weights using the five methods that have been described in §3 and §4. In particular, the best constant edge weight we find by solving the SDP (31) is -0.2030, which is outside the range (24) derived from the diagonally dominant condition (23). The smallest value for the diagonally dominant condition is given by the max-degree weight (25), which is -0.1251.

Table 1 shows the guaranteed convergence rates $\eta(W)$ obtained with different weight selection methods. These show that the max-degree has the largest value of η, with Metropolis and best constant about the same and smaller. The optimal symmetric and nonsymmetric weights (obtained by solving SDPs) give even faster convergence. (The nonsymmetric weight matrix W found by solving the SDP (31) has two positive off-diagonal entries; see the comment in the paragraph after equation (9).) Of course it must be remembered that η is only a guaranteed bound on the convergence rate, so small differences in η (such as between the optimal symmetric and optimal nonsymmetric weights) probably have no meaning in terms of actual convergence.

Figure 2 shows the objective values of the algorithm (5) using the five different weight matrices, all starting with the initial condition $x_i(0) = 0$ for all i. The plot shows that in this case the bound η does predict the actual convergence rate; in particular, the convergence with the optimal weights is substantially faster than, for example, the max-degree weights. This is frequently, but not always, the case.

6 Extensions

In this section, we extend the main results obtained in previous sections to optimization problems with more complex constraints.

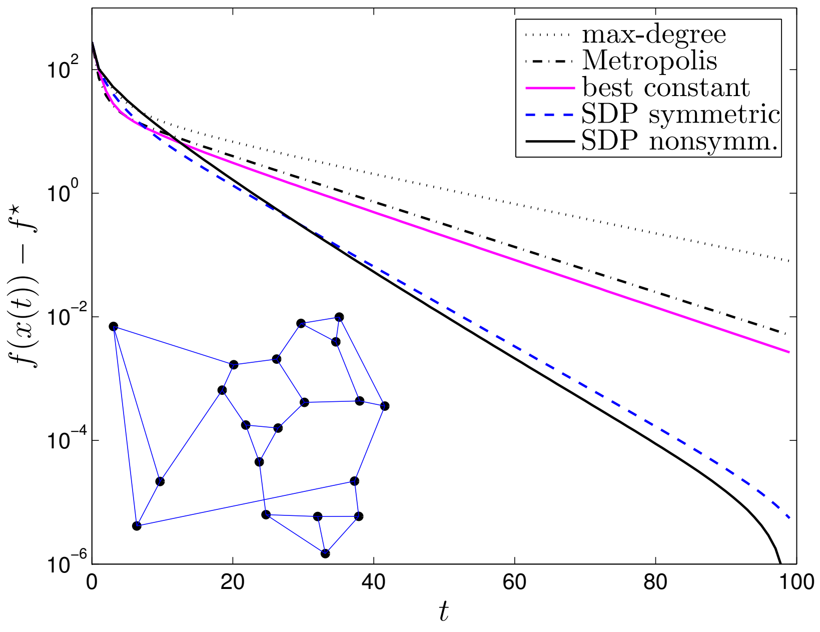

Figure 2: Objective function values $f(x(t)) - f^\star$ versus iteration number t, with five different weight matrices and the common initial condition $x(0) = 0$.

6.1 Problems with general equality constraints

We consider the following optimization problem with general linear equality constraints:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b, \end{array} \tag{33}$$

where $x \in \mathbf{R}^n$ is the variable and $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$ are given parameters. Without loss of generality, we assume that $m < n$ and A is full rank. We assume that the function $f(x)$ is strictly convex and

$$L \preceq \nabla^2 f(x) \preceq U$$

for some symmetric positive definite matrices L and U. (In fact, the above partial ordering only needs to hold on the nullspace of A, *i.e.*, $\{x \in \mathbf{R}^n \mid Ax = 0\}$.) Here we do not explicitly assume additional structure such as those imposed by a graph; we only note that the sparsity pattern of W could depend on the sparsity pattern of $\nabla^2 f(x)$. A point $x^\star$ is optimal for (33) if and only if there is a $v^\star \in \mathbf{R}^m$ such that

$$Ax^\star = b, \qquad \nabla f(x^\star) = A^T v^\star.$$

We consider solving the problem (33) using the weighted gradient descent algorithm (5). Here the feasibility and fixed point conditions translate into the following constraints on the weight matrix W:

$$AW = 0, \qquad WA^T = 0, \tag{34}$$

which correspond to conditions (7) and (8) respectively. The rows of A form the (left and right) eigenspace of W associated with the eigenvalue zero, which has multiplicity m. The additional sufficient condition for monotonic convergence becomes (*cf.* condition(11))

$$\lambda_{n-m}(V) = \lambda_{n-m}\left(L^{1/2}(W + W^T - W^T U W)L^{1/2}\right) > 0$$

which can be expressed as the strict LMI condition (*cf.* condition (14))

$$\begin{bmatrix} W + W^T + A^T(AA^T)^{-1}A & W \\ W^T & U^{-1} \end{bmatrix} \succ 0.$$

(As before, the convergence condition doesn't rely on L.) In this case, the guaranteed convergence rate is given by $\eta(W) = 1 - \lambda_{n-m}(V)$. Similarly, for optimization-based weight design, we only need to carry out the following replacements in the SDPs (30) and (31):

$$\begin{aligned} \frac{1}{n}\mathbf{1}\mathbf{1}^T \quad &\leftrightarrow \quad A^T(AA^T)^{-1}A \\ \frac{1}{\mathbf{1}^T L^{-1}\mathbf{1}} L^{-1}\mathbf{1}\mathbf{1}^T L^{-1} \quad &\leftrightarrow \quad L^{-1}A^T(AL^{-1}A^T)^{-1}AL^{-1}. \end{aligned}$$

In this case, however, it is difficult to come up with simple methods to select the weight matrix, like those given in §3. While the diagonally dominant condition (23) is still useful, it is difficult to directly construct weight matrices satisfying the eigenspace conditions in (34).

6.2 Problems with vector variables at each node

We consider a modified version of problem (1),

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n f_i(x_i) \\ \text{subject to} & \sum_{i=1}^n x_i = b, \end{array} \tag{35}$$

where each x_i is an m dimensional vector, *i.e.*, $x_i = (x_i^1, \ldots, x_i^m)$ for $i = 1, \ldots, n$. Here we do impose the local information structure given by the graph. In this case, We assume the following condition on the Hessians of the functions f_i

$$L_i \preceq \nabla^2 f(x_i) \preceq U_i, \quad i = 1, \ldots, n,$$

where L_i and U_i are positive definite matrices.

The center-free algorithm now takes the form

$$x_i(t+1) = x_i(t) + \sum_{j \in \mathcal{N}_i} W_{ij}\left(\nabla f_i(x_i(t)) - \nabla f_j(x_j(t))\right) \tag{36}$$

where $W_{ij} \in \mathbf{R}^{m \times m}$ is the weight matrix between two adjacent nodes i and j. The n by n blocks of W (each with size m by m) are either zero (if the corresponding two nodes are not connected) or a full m by m matrix (for two adjacent nodes). A simplified version of (36) has also been suggested in [HSS80], where the weight matrix W_{ij} between two adjacent node is diagonal.

The convergence condition and optimization-based weight design can be obtained by applying the results in §6.1. Here $x = (x_1^T, \ldots, x_n^T)$ is an augmented vector of length mn, and the equality constraint can be written as $Ax = b$, where $A = [I_m, \ldots, I_m]$ and I_m is the m-dimensional identity matrix. The weight matrix W should satisfy the eigenspace conditions (34). The lower and upper bounding matrices are block diagonal: $L = \mathbf{diag}(L_1, \ldots, L_m)$, $U = \mathbf{diag}(U_1, \ldots, U_n)$.

7 Conclusions

We have considered a class of distributed gradient algorithms for optimally allocate a fixed amount of resource over a network, to minimize the sum of the convex cost functions at each node. In these algorithms, each node updates the amount of its local resource in proportion to the differences between the marginal costs of itself and its neighbors. We focused on how to choose the proportional weights on the edges to make the algorithm converge, and how to optimize the weights (scaling of the gradient method) to get fast convergence. We should make several comments about the algorithm and the methods for selecting the weights.

First, the algorithm is designed for distributed implementation, with a simple protocol between each pair of connected nodes. In general, it is not competitive with off-line centralized optimization algorithms, such as BFGS methods, or Newton methods. These methods generally give faster (and sometimes much faster) convergence than the center-free algorithm described here; but on the other hand, they cannot be implemented in a simple, fully distributed way.

The second point concerns the cost of computing the optimal weights in the SDP-based weight selection method. While these weights evidently yield faster convergence of the method (compared to, say, a maximum degree or Metropolis choice of weights), it requires real computation, *i.e.*, the solution of an SDP. Solving the SDP to find the optimal weights often involves substantially *more* computational effort than solving the original resource allocation problem, so if the resource allocation problem is to be solved just once or a few times (say), this extra cost is certainly not justified. On the other hand, if we are to solve the resource allocation problem multiple times, on a network with the same topology and lower and upper bounds, with (say) different initial conditions or different cost functions, then the extra effort of solving an SDP to find the optimal weights can be justified.

In contrast, the Metropolis methods (which are based only on local information) are well suited for distributed weight selection, and can potentially be used with time-varying network topologies. One can imagine that at a slower time scale, nodes can join or leave the network, carrying in or out their current share of total resource. The algorithms presented in this paper, with the Metropolis weights determined in real-time, can run at a faster time scale to track the optimal solution of each network configuration.

Acknowledgment

We thank Professor Paul Tseng for his valuable comments that helped us to improve the presentation of this paper.

References

[AH60] K. J. Arrow and L. Hurwicz. Decentralization and computation in resource allocation. In R. W. Pfouts, editor, *Essays in Economics and Econometrics*, pages 34–104. University of North Carolina Press, Chapel Hill, NC, 1960.

[Baz96] D. El Baz. Asynchronous gradient algorithms for a class of convex separable network flow problems. *Computational Optimization and Applications*, 5:187–205, 1996.

[BDX03] S. Boyd, P. Diaconis, and L. Xiao. Fastest mixing Markov chain on a graph. Submitted to *SIAM Review*, problems and techniques section, February 2003. Available at `www.stanford.edu/~boyd/fmmc.html`.

[Ber98] D. P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, 1998.

[Ber99] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1999.

[BT89] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation*. Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[BV03] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2003. Available at `www.stanford.edu/~boyd/cvxbook.html`.

[DSC98] P. Diaconis and L. Saloff-Coste. What do we know about the Metropolis algorithms. *Journal of Computer and System Sciences*, 57:20–36, 1998.

[GR01] C. Godsil and G. Royle. *Algebraic Graph Theory*, volume 207 of *Graduate Texts in Mathematics*. Springer, 2001.

[Hea69] G. M. Heal. Planning without prices. *The Review of Economic Studies*, 36(3):347–362, July 1969.

[HJ85] R. A. Horn and C. A. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[HSS80] Y. C. Ho, L. Servi, and R. Suri. A class of center-free resource allocation algorithms. *Large Scale Systems*, 1:51–62, 1980.

[Hur73] L. Hurwicz. The design of mechanisms for resource allocation. *The American Economic Review*, 63(2):1–30, May 1973.

[KS89] J. F. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Transactions on Computers*, 38(5):705–717, May 1989.

[LT94] Z.-Q. Luo and P. Tseng. On the rate of convergence of a distributed asynchronous routing algorithm. *IEEE Transactions on Automatic Control*, 39(5):1123–1129, May 1994.

[Mer94] R. Merris. Laplacian matrices of graphs: a survey. *Linear Algebra and Its Applications*, 197:143–176, 1994.

[MRR$^+$53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1092, 1953.

[NN94] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming.* SIAM Studies in Applied Mathematics. SIAM, 1994.

[Roc84] R. T. Rockafellar. *Network Flows and Monotropic Optimization.* John Wiley & Sons, New York, 1984.

[Ser80] L. D. Servi. Electrical networks and resource allocation algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 10(12):841–848, December 1980.

[TB86] J. N. Tsitsiklis and D. P. Bertsekas. Distributed asynchronous optimal routing in data networks. *IEEE Transactions on Automatic Control*, 31(4):325–332, April 1986.

[TBA86] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, September 1986.

[VB96] L. Vandenberghe and S. Boyd. Semidefinite programming. *Siam Review*, 38(1):49–95, 1996.

[WSV00] H. Wolkowicz, R. Saigal, and L. Vandengerghe, editors. *Handbook of Semidefinite Programming, Theory, Algorithms, and Applications.* Kluwer Academic Publishers, 2000.

[XB03] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. Submitted to *Systems and Control Letters*, March 2003. Available on the Internet at `www.stanford.edu/~boyd/fastavg.html`.

[Ye97] Y. Ye. *Interior Point Algorithms: Theory and Analysis.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, New York, 1997.